1. 请简洁描述 MySQL 中 InnoDB 支持的四种事务隔离级别名称，以及逐级之间的区别？

SQL 标准定义的四个隔离级别为：

read uncommited ：读到未提交数据

read committed：脏读，不可重复读

repeatable read：可重读

serializable ：串行事物

2. 在 MySQL 中 ENUM 的用法是什么？

ENUM 是一个字符串对象，用于指定一组预定义的值，并可在创建表时使用。

SQL 语法如下：

```
Create table size(name ENUM('Smail,'Medium','Large');
```

3. CHAR 和 VARCHAR 的区别？

CHAR 和 VARCHAR 类型在存储和检索方面有所不同。

CHAR 列长度固定为创建表时声明的长度，长度值范围是 1 到 255。

当 CHAR 值被存储时，它们被用空格填充到特定长度，检索 CHAR 值时需删除尾随空格。

4. 列的字符串类型可以是什么？

字符串类型是：

VISIT…

SET

BLOB

ENUM

CHAR

TEXT

VARCHAR

5. MySQL 中使用什么存储引擎？

存储引擎称为表类型，数据使用各种技术存储在文件中。

技术涉及：

Storage mechanism

Locking levels

Indexing

Capabilities and functions.

6. TIMESTAMP 在 UPDATE CURRENT_TIMESTAMP 数据类型上做什么？

创建表时 TIMESTAMP 列用 Zero 更新。只要表中的其他字段发生更改，UPDATE CURRENT_TIMESTAMP 修饰符就将时间戳字段更新为当前时间。

7. 主键和候选键有什么区别？

表格的每一行都由主键唯一标识，一个表只有一个主键。

主键也是候选键。按照惯例，候选键可以被指定为主键，并且可以用于任何外键引用。

8. MySQL 数据库服务器性能分析的方法命令有哪些?

Show status 一些值得监控的变量值： Bytes*received 和 Bytes*sent 和服务器之间来往的流量。 Com **服务器正在执行的命令。 Created** 在查询执行期限间创建的临时表和文件。 Handler **存储引擎操作。 Select** 不同类型的联接执行计划。 Sort_* 几种排序信息。 Show session status like ‘Select’; Show profiles SET profiling=1; Show profiles\G Show profile;

9. LIKE 和 REGEXP 操作有什么区别？

LIKE 和 REGEXP 运算符用于表示 ^ 和%。

```
SELECT * FROM <tablename> WHERE * REGEXP "^b";
SELECT * FROM <tablename> WHERE * LIKE "%b";
```

10. BLOB 和 TEXT 有什么区别？

BLOB

BLOB 是一个二进制对象，可以容纳可变数量的数据。有四种类型的 BLOB

- TINYBLOB
- BLOB
- MEDIUMBLOB 和

- LONGBLOB

它们只能在所能容纳价值的最大长度上有所不同。

TEXT

TEXT 是一个不区分大小写的 BLOB。四种 TEXT 类型

- TINYTEXT
- TEXT
- MEDIUMTEXT 和
- LONGTEXT

它们对应于四种 BLOB 类型，并具有相同的最大长度和存储要求。

BLOB 和 TEXT 类型之间的唯一区别在于对 BLOB 值进行排序和比较时区分大小写，对 TEXT 值不区分大小写。

11. 数据库的三范式？

第一范式：数据库表的每一个字段都是不可分割的。

第二范式：数据库表中的非主属性只依赖于主键。

第三范式：不存在非主属性对关键字的传递函数依赖关系。

12. MySQL 表中允许有多少个 TRIGGERS？

在 MySQL 表中允许有六个触发器，如下：

BEFORE INSERT

AFTER INSERT

BEFORE UPDATE

AFTER UPDATE

BEFORE DELETE and

AFTER DELETE

13. 什么是通用 SQL 函数？

数学函数

- Abs（num）求绝对值
- floor（num）向下取整
- ceil（num）向上取整

字符串函数

- insert (s1,index,length,s2) 替换函数
    - S1 表示被替换的字符串

    - s2 表示将要替换的字符串
    - Index 表示被替换的位置，从 1 开始
    - Lebgth 表示被替换的长度
- upper（str），ucase（str）将字母改为大写
- lower（str），lcase（str）将字母改为小写
- left（str，length）返回 str 字符串的前 length 个字符
- right（str，length）返回 str 字符串的后 length 个字符
- substring（str，index，length）返回 str 字符串从 index 位开始长度为 length 个字符（index 从 1 开始）
- reverse（str）将 str 字符串倒序输出

日期函数

- curdate（ ）、current_date( ) 获取当前日期
- curtime（ ）、current_time( ) 获取当前日期
- now（ ）获取当前日期和时间
- datediff（d1、d2）d1 和 d2 之间的天数差
- adddate（date，num）返回 date 日期开始，之后 num 天的日期
- subdate（date，num）返回 date 日期开始，之前 num 天的日期

聚合函数

- Count（字段）根据某个字段统计总记录数（当前数据库保存到多少条数据）
- sum（字段）计算某个字段的数值总和
- avg（字段）计算某个字段的数值的平均值
- Max（字段）、min（字段）求某个字段最大或最小值

14. MySQL 中有哪几种锁？

MyISAM 支持表锁，InnoDB 支持表锁和行锁，默认为行锁。

表级锁：开销小，加锁快，不会出现死锁。锁定粒度大，发生锁冲突的概率最高，并发量最低。

行级锁：开销大，加锁慢，会出现死锁。锁力度小，发生锁冲突的概率小，并发度最高。

15. MySQL 数据优化。

- **优化数据类型**
    - 避免使用 NULL，NULL 需要特殊处理，大多数时候应该使用 NOT NULL，或者使用一个特殊的值，如 0，-1 作为默认值。

    - 仅可能使用更小的字段，MySQL 从磁盘读取数据后是存储到内存中的，然后使用 cpu 周期和磁盘 I/O 读取它，这意味着越小的数据类型占用的空间越小.

- **小心字符集转换**

    - 客户端或应用程序使用的字符集可能和表本身的字符集不一样，这需要 MySQL 在运行过程中隐含地进行转换，此外，要确定字符集如 UTF-8 是否支持多字节字符，因此它们需要更多的存储空间。

- *_优化 count(my*col) 和 count(*)*

- **优化子查询**

    - 遇到子查询时，MySQL 查询优化引擎并不是总是最有效的，这就是为什么经常将子查询转换为连接查询的原因了，优化器已经能够正确处理连接查询了，当然要注意的一点是，确保连接表（第二个表）的连接列是有索引的，在第一个表上 MySQL 通常会相对于第二个表的查询子集进行一次全表扫描，这是嵌套循环算法的一部分。

- **优化 UNION**

    - 在跨多个不同的数据库时使用 UNION 是一个有趣的优化方法，UNION 从两个互不关联的表中返回数据，这就意味着不会出现重复的行，同时也必须对数据进行排序，我们知道排序是非常耗费资源的，特别是对大表的排序。
    - UNION ALL 可以大大加快速度，如果你已经知道你的数据不会包括重复行，或者你不在乎是否会出现重复的行，在这两种情况下使用

UNION ALL 更适合。此外，还可以在应用程序逻辑中采用某些方法避免出现重复的行，这样 UNION ALL 和 UNION 返回的结果都是一样的，但 UNION ALL 不会进行排序。

16. MySQL 的关键字。

添加索引：

```
alter table tableName add 索引（索引字段）
```

主键：primary key

唯一：unique

全局：fulltext

普通：index

多列： index index_name

页级: 引擎 BDB。次锁定相邻的一组记录。

表级: 引擎 MyISAM ， 理解为锁住整个表，可以同时读，写不行。 行级: 引擎 INNODB ， 单独的一行记录加锁，对指定的记录进行加锁，这样其它进程还是可以对同一个表中的其它记录进行操作。 表级锁速度快，但冲突多，行级冲突少，但速度慢。

17. 存储引擎。

存储引擎说白了就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等技术的实现方法。

- MyISAM：这种引擎是 mysql 最早提供的。这种引擎又可以分为静态 MyISAM、动态 MyISAM 和压缩 MyISAM 三种：
    - 静态 MyISAM：如果数据表中的各数据列的长度都是预先固定好的，服务器将自动选择这种表类型。因为数据表中每一条记录所占用的空间都是一样的，所以这种表存取和更新的效率非常高。当数据受损时，恢复工作也比较容易做。
    - 动态 MyISAM：如果数据表中出现 varchar、text 或 BLOB 字段时，服务器将自动选择这种表类型。相对于静态 MyISAM，这种表存储空间比较小，但由于每条记录的长度不一，所以多次修改数据后，数据表中的数据就可能离散的存储在内存中，进而导致执行效率下降。同时，内存中也可能会出现很多碎片。因此，这种类型的表要经常用 optimize table 命令或优化工具来进行碎片整理。
    - 压缩 MyISAM：以上说到的两种类型的表都可以用 myisamchk 工具压缩。这种类型的表进一步减小了占用的存储，但是这种表压缩之后不能再被修改。另外，因为是压缩数据，所以这种表在读取的时候要先时行解压缩。

但是，不管是何种 MyISAM 表，目前它都不支持事务，行级锁和外键约束的功能。

- MyISAM Merge 引擎：这种类型是 MyISAM 类型的一种变种。合并表是将几个相同的 MyISAM 表合并为一个虚表。常应用于日志和数据仓库。

- InnoDB：InnoDB 表类型可以看作是对 MyISAM 的进一步更新产品，它提供了事务、行级锁机制和外键约束的功能。
- memory(heap)：这种类型的数据表只存在于内存中。它使用散列索引，所以数据的存取速度非常快。因为是存在于内存中，所以这种类型常应用于临时表中。
- archive：这种类型只支持 select 和 insert 语句，而且不支持索引。
- Desc[ribe] tablename：查看数据表的结构。
- show engines：命令可以显示当前数据库支持的存储引擎情况。

18. 数据库备份。

必须要在未登录状态下

- 导出整个数据库

```
mysqldump -u 用户名 -p 数据库名 > 导出的文件名
```

- 导出一个表

```
mysqldump -u 用户名 -p 数据库名 表名> 导出的文件名
```

- 导出一个数据库结构

```
mysqldump -u dbuser -p -d --add-drop-table
dbname >d:/dbname_db.sql
```

-d 没有数据 --add-drop-table 在每个 create 语句之前增加一个 drop table

19. truncate delete drop 的区别。

drop(DDL 语句)：是不可逆操作，会将表所占用空间全部释放掉；

truncate(DDL 语句)：只针对于删除表的操作，在删除过程中不会激活与表有关的删除触发器并且不会把删除记录放在日志中；当表被 truncate 后，这个表和索引会恢复到初始大小；

delete(DML 语句)：可以删除表也可以删除行，但是删除记录会被计入日志保存，而且表空间大小不会恢复到原来；

执行速度：drop>truncate>delete。

20. Redis 是什么？两句话做一下概括。

是一个完全开源免费的 key-value 内存数据库 2. 通常被认为是一个数据结构服务器，主要是因为其有着丰富的数据结构 strings、map、 list、sets、 sorted sets。

- Redis 使用最佳方式是全部数据 in-memory。
- Redis 更多场景是作为 Memcached 的替代者来使用。
- 当需要除 key/value 之外的更多数据类型支持时，使用 Redis 更合适。
- 当存储的数据不能被剔除时，使用 Redis 更合适。

21. Redis（管道，哈希）。

- Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list，set，zset，hash 等数据结构的存储。
- Redis 支持数据的备份，即 master-slave 模式的数据备份。
- Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。

22. Redis 实现原理或机制。

Redis 是一个 key-value 存储系统。和 Memcached 类似，但是解决了断电后数据完全丢失的情况，而且她支持更多无化的 value 类型，除了和 string 外，还支持 lists（链表）、sets（集合）和 zsets（有序集合）几种数据类型。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。

Redis 是一种基于客户端 - 服务端模型以及请求 / 响应协议的 TCP 服务。这意味着通常情况下一个请求会遵循以下步骤：

客户端向服务端发送一个查询请求，并监听 Socket 返回，通常是以阻塞模式，等待服务端响应。服务端处理命令，并将结果返回给客户端。

在服务端未响应时，客户端可以继续向服务端发送请求，并最终一次性读取所有服务端的响应。

Redis 管道技术最显著的优势是提高了 Redis 服务的性能。

分区是分割数据到多个 Redis 实例的处理过程，因此每个实例只保存 key 的一个子集。

通过利用多台计算机内存的和值，允许我们构造更大的数据库。

通过多核和多台计算机，允许我们扩展计算能力；通过多台计算机和网络适配器，允许我们扩展网络带宽。

Redis 的一些特性在分区方面表现的不是很好：

涉及多个 key 的操作通常是不被支持的。举例来说，当两个 set 映射到不同的 Redis 实例上时，你就不能对这两个 set 执行交集操作。

涉及多个 key 的 Redis 事务不能使用。

当使用分区时，数据处理较为复杂，比如你需要处理多个 rdb/aof 文件，并且从多个实例和主机备份持久化文件。

增加或删除容量也比较复杂。Redis 集群大多数支持在运行时增加、删除节点的透明数据平衡的能力，但是类似于客户端分区、代理等其他系统则不支持这项特性。然而，一种叫做 presharding 的技术对此是有帮助的。

23. Redis 有两种类型分区。

最简单的分区方式是按范围分区，就是映射一定范围的对象到特定的 Redis 实例。

比如，ID 从 0 到 10000 的用户会保存到实例 R0，ID 从 10001 到 20000 的用户会保存到 R1，以此类推。

这种方式是可行的，并且在实际中使用，不足就是要有一个区间范围到实例的映射表。这个表要被管理，同时还需要各 种对象的映射表，通常对 Redis 来说并非是好的方法。

哈希分区：另外一种分区方法是 hash 分区。这对任何 key 都适用，也无需是 object_name: 这种形式，像下面描述的一样简单：

用一个 hash 函数将 key 转换为一个数字，比如使用 crc32 hash 函数。对 key foobar 执行 crc32(foobar) 会输出类似 93024922 的整数。

对这个整数取模，将其转化为 0-3 之间的数字，就可以将这个整数映射到 4 个 Redis 实例中的一个了。93024922 % 4 = 2，就是说 key foobar 应该被存到 R2 实例中。注意：取模操作是取除的余数，通常在多种编程语言中用 % 操作符实现。

实际上，上面的集群模式还存在两个问题：

1. 扩容问题：

   因为使用了一致性哈稀进行分片，那么不同的 key 分布到不同的 Redis-Server 上，当我们需要扩容时，需要增加机器到分片列表中，这时候会使得同样的 key 算出来落到跟原来不同的机器上，这样如果要取某一个值，会出现取不到的情况，对于这种情况，Redis 的作者提出了一种名为 Pre-Sharding 的方式：

   Pre-Sharding 方法是将每一个台物理机上，运行多个不同断口的 Redis 实例，假如有三个物理机，每个物理机运行三个 Redis 实际，那么我们的分片列表中实际有 9 个 Redis 实例，当我们需要扩容时，增加一台物理机，步骤如下：

   1. 在新的物理机上运行 Redis-Server；
   2. 该 Redis-Server 从属于 (slaveof) 分片列表中的某一 Redis-Server（假设叫 RedisA）；

3. 等主从复制（Replication）完成后，将客户端分片列表中 RedisA 的 IP 和端口改为新物理机上 Redis-Server 的 IP 和端口；
4. 停止 RedisA。

这样相当于将某一 Redis-Server 转移到了一台新机器上。Prd-Sharding 实际上是一种在线扩容的办法，但还是很依赖 Redis 本身的复制功能的，如果主库快照数据文件过大，这个复制的过程也会很久，同时会给主库带来压力。所以做这个拆分的过程最好选择为业务访问低峰时段进行。

2. 单点故障问题：

还是用到 Redis 主从复制的功能，两台物理主机上分别都运行有 Redis-Server，其中一个 Redis-Server 是另一个的从库，采用双机热备技术，客户端通过虚拟 IP 访问主库的物理 IP，当主库宕机时，切换到从库的物理 IP。只是事后修复主库时，应该将之前的从库改为主库（使用命令 slaveof no one），主库变为其从库（使命令 slaveof IP PORT），这样才能保证修复期间新增数据的一致性。

24. 什么是 MongoDB。

非关系型数据库（NoSql),Mongo DB 很好的实现了面向对象的思想（OO 思想），在 Mongo DB 中 每一条记录都是一个 Document 对象。Mongo DB 最大的优势在于所有的数据持久操作都无需开发人员手动编写 SQL 语句，直接调用方法就可以轻松的实现 CRUD 操作.

25. MongoDB 特点。

高性能、易部署、易使用，存储数据非常方便。主要功能特性有：

面向集合存储，易存储对象类型的数据。

模式自由。

支持动态查询。

支持完全索引，包含内部对象。

支持查询。

支持复制和故障恢复。

使用高效的二进制数据存储，包括大型对象（如视频等）。

自动处理碎片，以支持云计算层次的扩展性

支持 Python，PHP，Ruby，Java，C，C#，Javascript，Perl 及 C++ 语言的驱动程序，社区中也提供了对 Erlang 及. NET 等平台的驱动程序。

文件存储格式为 BSON（一种 JSON 的扩展）。

可通过网络访问。

26. MongoDB 的功能。

面向集合的存储：适合存储对象及 JSON 形式的数据。

动态查询：Mongo 支持丰富的查询表达式。查询指令使用 JSON 形式的标记，可轻易查询文档中内嵌的对象及数组。

完整的索引支持：包括文档内嵌对象及数组。Mongo 的查询优化器会分析查询表达式，并生成一个高效的查询计划。

查询监视：Mongo 包含一个监视工具用于分析数据库操作的性能。

复制及自动故障转移：Mongo 数据库支持服务器之间的数据复制，支持主 - 从模式及服务器之间的相互复制。复制的主要目标是提供冗余及自动故障转移。

高效的传统存储方式：支持二进制数据及大型对象（如照片或图片）

自动分片以支持云级别的伸缩性：自动分片功能支持水平的数据库集群，可动态添加额外的机器。

27. MongoDB 的适用场景。

网站数据：Mongo 非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

缓存：由于性能很高，Mongo 也适合作为信息基础设施的缓存层。在系统重启之后，由 Mongo 搭建的持久化缓存层可以避免下层的数据源 过载。

大尺寸，低价值的数据：使用传统的关系型数据库存储一些数据时可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储。

高伸缩性的场景：Mongo 非常适合由数十或数百台服务器组成的数据库。Mongo 的路线图中已经包含对 MapReduce 引擎的内置支持。

用于对象及 JSON 数据的存储：Mongo 的 BSON 数据格式非常适合文档化格式的存储及查询。

28. Redis、memcache、MongoDB 对比。

mongodb 和 memcached 不是一个范畴内的东西。mongodb 是文档型的非关系型数据库，其优势在于查询功能比较强大，能存储海量数据。

和 memcached 更为接近的是 Redis。它们都是内存型数据库，数据保存在内存中，通过 tcp 直接存取，优势是速度快，并发高，缺点是数据类型有限，查询功能不强，一般用作缓存。

1. 性能

   Redis 和 memcache 差不多，要大于 mongodb。

2. 操作的便利性

   memcache 数据结构单一。

   Redis 丰富一些，数据操作方面，Redis 更好一些，较少的网络 IO 次数。

   mongodb 支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富。

3. 内存空间的大小和数据量的大小

   Redis 在 2.0 版本后增加了自己的 VM 特性，突破物理内存的限制；可以对 key value 设置过期时间（类似 memcache）。

memcache 可以修改最大可用内存，采用 LRU 算法。

mongoDB 适合大数据量的存储，依赖操作系统 VM 做内存管理，吃内存也比较厉害，服务不要和别的服务在一起。

4. 可用性（单点问题）

Redis 对于单点问题，依赖客户端来实现分布式读写；主从复制时，每次从节点重新连接主节点都要依赖整个快照，无增量复制，因性能和效率问题，所以单点问题比较复杂；不支持自动 sharding，需要依赖程序设定一致 hash 机制。一种替代方案是，不用 Redis 本身的复制机制，采用自己做主动复制（多份存储），或者改成增量复制的方式（需要自己实现），一致性问题和性能的权衡。

Memcache 本身没有数据冗余机制，也没必要；对于故障预防，采用依赖成熟的 hash 或者环状的算法，解决单点故障引起的抖动问题。

mongoDB 支持 master-slave,replicaset（内部采用 paxos 选举算法，自动故障恢复）,auto sharding 机制，对客户端屏蔽了故障转移和切分机制。

5. 可靠性（持久化）

对于数据持久化和数据恢复，Redis 支持（快照、AOF）：依赖快照进行持久化，aof 增强了可靠性的同时，对性能有所影响。

memcache 不支持，通常用在做缓存，提升性能；

MongoDB 从 1.8 版本开始采用 binlog 方式支持持久化的可靠性。

6. 数据一致性（事务支持）

   Memcache 在并发场景下，用 cas 保证一致性。

   Redis 事务支持比较弱，只能保证事务中的每个操作连续执行。

   mongoDB 不支持事务。

7. 数据分析

   mongoDB 内置了数据分析的功能（mapreduce），其他不支持。

8. 应用场景

   Redis：数据量较小的更性能操作和运算上。

   memcache：用于在动态系统中减少数据库负载，提升性能；做缓存，提高性能（适合读多写少，对于数据量比较大，可以采用 sharding）。

   MongoDB: 主要解决海量数据的访问效率问题。

29. Redis 有什么用？只有了解了它有哪些特性，我们在用的时候才能扬长避短，为我们所用。

1. **速度快：**使用标准 C 写，所有数据都在内存中完成，读写速度分别达到 10 万 / 20 万。
2. **持久化：**对数据的更新采用 Copy-on-write 技术，可以异步地保存到磁盘上，主要有两种策略，一是根据时间，更新次数的快照（save 300 10 ）二是基于语句追加方式（Append-only file，aof）。
3. **自动操作：**对不同数据类型的操作都是自动的，很安全。

4. 快速的主 -- 从复制，官方提供了一个数据，Slave 在 21 秒即完成了对 Amazon 网站 10G key set 的复制。
5. **Sharding 技术：** 很容易将数据分布到多个 Redis 实例中，数据库的扩展是个永恒的话题，在关系型数据库中，主要是以添加硬件、以分区为主要技术形式的纵向扩展解决了很多的应用场景，但随着 web2.0、移动互联网、云计算等应用的兴起，这种扩展模式已经不太适合了，所以近年来，像采用主从配置、数据库复制形式的，Sharding 这种技术把负载分布到多个特理节点上去的横向扩展方式用处越来越多。